Efficient Agent-Based Models for Non-Genomic Evolution

Nachi Gupta
Oxford Univ., Computing Lab
Numerical Analysis Group
Wolfson Building, Parks Road
Oxford, OX1 3QD, UK
nachi@comlab.ox.ac.uk

Adrian Agogino
UC Santa Cruz
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035, USA
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-4
Moffett Field, CA 94035, USA
ktumer@mail.arc.nasa.gov

ABSTRACT

Modeling dynamical systems composed of aggregations of primitive proteins is critical to the field of astrobiological science, which studies early evolutionary structures dealing with the origins of life. Current theories on the emergence of early life focus either on RNA-world models [2, 3] or protein world models [4, 5]. Traditional models from either model are generally either too slow to converge or too simplified to provide good tools for exploring the trade-offs in the early stages of the emergence of life. This paper focuses on protein-world models and discusses how to model protein aggregations through a utility based multi-agent system. We define agents to control specific properties of a given set of proteins. These properties determine the dynamics of the system, such as the ability for proteins to join or split apart, while additional properties determine the aggregation's fitness as a viable primitive cell. We show that over a wide range of starting conditions, there are mechanisms that allow protein aggregations to achieve high values of overall fitness. In addition through the use of agent-specific utilities that remain aligned with the overall global utility, we are able to reach these conclusions with 50 times fewer learning steps.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence— *Multiagent systems*

General Terms

Algorithms, Performance

Keywords

Multiagent Systems, Non-genomic Evolution, Reinforcement Learning

1. INTRODUCTION

In this paper we show how multi-agent systems can make an important contribution to the field of astrobiology - the study of primitive ecosystems that occurred early in Earth's history and may be present of other planets. We focus on the subfield of "non-genomic evolution," involving the evolution of protein ecologies that do not contain nucleic acids (DNA and RNA) which form the basis of modern evolution. Protein ecologies are important since simple proteins are easier to make through natural processes than nucleic acids and it is thought that life on Earth might have started with these ecologies. The significant feature of protein ecologies is that unlike nucleic acids, proteins cannot directly replicate themselves. As a result, non-genomic evolution is concerned with the global behavior of the entire protein ecology and its ability as a whole to form new protein ecologies. It is therefore a good domain to use multi-agent systems to model the interactions between proteins and the emergence of specific protein ecologies.

The study of non-genomic evolution seeks to find the initial conditions and mechanisms that must be present to form self-sustaining protein ecologies [6]. Unfortunately with current knowledge it is impossible to directly simulate the evolution of a protein ecology. Direct evolution of low-level chemical processes has limited promise, because of the computational difficulties in creating a simulation that would encompass a full planetary ecology. In principle, feasible simulations can be obtained by modeling the higher level dynamics of protein interactions that map a protein's chemical structure (amino-acid sequences) to probability distributions over possible functions. However, because at present few of these mappings are known, a meaningful simulation at this level is not possible. As a consequence non-genomic evolution is currently analyzed using "plausible" models for the dynamics of the protein ecology [4, 5]. These simplified dynamical models are then run and are evaluated based on a global utility function. This global utility abstractly measures the value of the resulting protein ecology. Unfortunately creating a model that leads to a good final ecology is laborious and extremely time comsuming.

This paper shows how learning agents can be used to improve this modeling process by having agents automatically learn parameters of protein ecology models that lead to high global fitness, allowing more complex and meaningful models than the ones currently used in the astrobiological community. In this approach agents choose properties of individual proteins that affect the dynamics of the system (such as a proteins ability to split other proteins) as well as protein

properties that are important in producing a viable ecology, such as the ability of an ecology to produce lipids needed for a cell membrane. In a continuous process, agents choose protein properties while the distribution and number of proteins is being updated through the dynamics of the ecology. Using reinforcement learning, the agents attempt to make choices that lead to high global fitness, determined by a given utility function evaluating the ecology's viability. In our experiments the mapping between the proteins' properties and the ecology's global utility is more complex, indirect and realistic than in the models previously used in the astrobiological community. As a consequence ecologies that are able to achieve high fitness provide a better example of mechanisms that can lead to self-sustaining protein ecologies.

Section 2 describes the underlying principles and previous work associated with non-genomic evolution. Section 3 mathematically describes the model used in this paper simulating the dynamics and evaluating ecologies of proteins. Section 4 shows how learning agents are used to optimize the properties of the proteins so that the entire system achieves high global fitness. Section 5.1 presents results from simulations of the system showing that ecologies with high global fitness can be found quickly. Section 6 then discusses the significance of these results to the field of astrobioligy, and provides future directions in which multi-agent systems can be used to benefit this study.

2. BACKGROUND AND PREVIOUS WORK

All current life is based on organisms replicating through information coded in their DNA (Deoxyribonucleic acid) or RNA (Ribonucleic acid). DNA and RNA are composed of sequences of nucleic acids, which can form base-pairs allowing replication of the sequence. Due to their ability to replicate, models of the origins of life based on DNA or RNA are the most popular models [3]. However, DNA and/or RNA based models have many difficulties [6]. Current organisms require a complex interaction between DNA and RNA to survive. Even hypothetical organisms based only on RNA ("RNA worlds") require complicated interactions between different types of RNA [2, 3]. In addition, the nucleic acid sequences which form DNA and RNA are difficult to make under the pre-biotic conditions that existed early in Earth's history.

Therefore, there is a growing interest in non-genomic evolution - the study of organisms that do not use nucleic acids. Instead of using nucleic acids, these organisms use only proteins. The non-genomic evolutionary model of primitive organisms has the advantage that the proteins are made of sequences of amino-acids, which have been shown to be made under conditions similar to those that existed in Earth's early history. However, since amino-acids cannot form base-pairs, unlike nucleic acids, proteins cannot be directly replicated (Figure 1). Therefore the field of non-genomic evolution focuses on global properties of protein ecologies to see if protocells (precursors to cells which have some properties of cells such as having a membrane, but do not have all cell functions or cell components) formed from these ecologies can produce new protocells [2, 6]. Understanding the combined behavior of the ecology is paramount in order to ascertain the potential place of nongenetic evolution in the history of life on Earth.

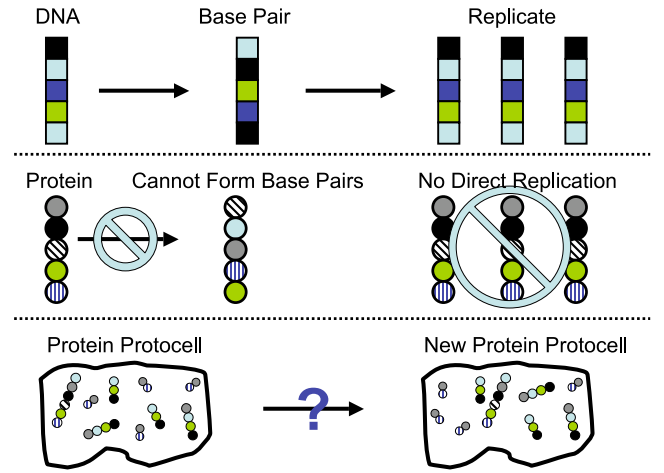While recent laboratory and computation progress has

Figure 1: Replication of Proteins. DNA and RNA can form base pairs and directly replicate their sequences. Proteins cannot form base pairs and therefore cannot directly replicate themselves. However as a system, a group of proteins forming a protocell may be able to replicate by creating other protocells with approximately the same characteristics.

been made in understanding self-replicating protein ecologies, we are still far from being able to precisely model the evolution of a protocell based on its constituent proteins [4, 5, 7, 8, 11]. As a result current models are greatly simplified, using high-level assumption about the dynamics of the system [6]. One of the few models for non-genomic evolution is described in New and Pohorille [6] and is based on the relationship between the length of the protein and its efficiency - the percentage of time it succeeds in performing operations on other proteins. In this model, proteins perform two operations: hydrolysis and ligation. Hydrolysis is the process of decomposing a protein, whereas ligation is the process of building a new protein by joining two proteins. In this model the efficiency of ligation and hydrolysis are based on Gaussians, with longer proteins generally being more efficient. The fitness evaluation of the protocell in this model is based on protein length with longer proteins being evaluated higher than proteins with shorter length. The computational experiments performed with this model show that protocells become more fit with time. However, in this model the fitness evaluation is only concerned with the efficiency of two functions while numerous other properties are required for a viable protocell, such as a protein's ability to produce cell-membrane building lipids.

In this paper we will expand on these results and develop a more complex model of non-genomic evolution with the help of a multi-agent system. This setup differs from previous ones in that efficiencies that the proteins attain at certain tasks is determined by the actions of learning agents, allowing many more degrees of freedom in the dynamics of the evolution. In addition the proteins have more functions besides their ability to break and build other proteins. Finally the global utility used in evaluating the fitness of the protocell takes into account the distribution of all the protein functions in the protocell.

3. THE PROTOCELL MODEL

The model of the primitive protocell used in this paper is based on different protein types performing different tasks within the protocell and contributing differently to the fitness of the protocell. In this model a protocell has a set of proteins of various types. The two most important types of proteins are combiner and breaker proteins. Combiner proteins take two other proteins and combine them into a single protein (ligation). Breaker proteins take a single other protein and break them into two separate proteins (hydrolysis). Other protein types have important functions such as lipid production for cell walls, energy production and energy transport. However in our model, these other proteins do not actively contribute to the dynamics of the system and are only important in evaluating the final viability of the protocell as defined by the system's global utility.

3.1 Global Utility

For a protocell to survive and thrive, it needs different types of proteins performing different tasks. Certain types of proteins are needed for building cell membranes, while others are needed for tasks such as energy production, energy transportation and chemical exchange. If a cell has a poor distribution of proteins, such as having no proteins for building cell membranes, it will be unlikely to survive and should receive a poor utility. Our model makes an abstraction over the different types of possible proteins, ignoring their low-level functions. We only assume that there are optimal numbers of each type of protein and we use an exponential decay to assign utility to off-optimal distributions.

To formally define the global utility of a protocell, let there be k possible types of proteins and let y_i be the desired amount of proteins of type i in the system, while x_i represents the current amount of proteins of type i present in the system. We choose the functional form $x_i e^{-\frac{x_i}{y_i}}$, which takes its optimal value of 1 when $x_i = y_i$, to describe how close we are to the desired number of proteins performing function k. The intuition behind this choice is that there is an optimal amount of a particular protein needed in the protocell. If there is less than that amount, the protocell would not function properly, and if there is more than that amount, the protein uses more resources than needed to perform its task.

The global utility for the protocell is simply the sum of these functions:

$$G(x, y) = \sum_{i=1}^{k} x_i e^{-\frac{x_i}{y_i}} , \qquad (1)$$

where x is the vector of the current amounts of each protein type and y is the vector of desired amounts of each protein type.

3.2 Protocell Dynamics

The dynamics of the protocell are determined by hydrolysis and ligation. During ligation, a "combiner protein" joins two proteins into a single protein. During hydrolysis, on the other hand, a protein is broken into two proteins. There are two crucial issues in both operations. First, the combiner or breaker protein has to choose the proteins on which it will act. Second the protein(s) resulting from the operation need(s) to inherit certain properties of their predecessors. These issues will be modeled through concepts of *specialization* and *clustering* respectively.

Based on current theory of non-genomic evolution, it is unlikely that combiner proteins in early protocells were specialized, i.e. they will combine two other proteins together regardless of what type they are [6]. However, it is likely that breaker proteins were somewhat specialized: There are some types of proteins which a particular breaker protein will not break. While the actual specialization dynamics are complex, we will model this specialization by assigning each breaker protein a single type of protein which it will never break. All other proteins will be broken by that protein.

When new proteins are created as a result of combining or breaking operations, they will be assigned types based on the proteins they came from. This assignment will be modeled through the principle of "protein type clusters." In this model we assume that proteins of the same type tend to congregate near each other, forming clusters. In our model each protein belongs to exactly one cluster, though multiple clusters may have proteins of similar types (Figure 2). When a new protein is created as a result of a breaking operation it will tend to go to a cluster of the same type as its parent (though not necessarily the same cluster). When a new protein is created based on combing two proteins, it will go to a cluster chosen based on the properties of the parent clusters. Details of these assignments are given in Section 4.
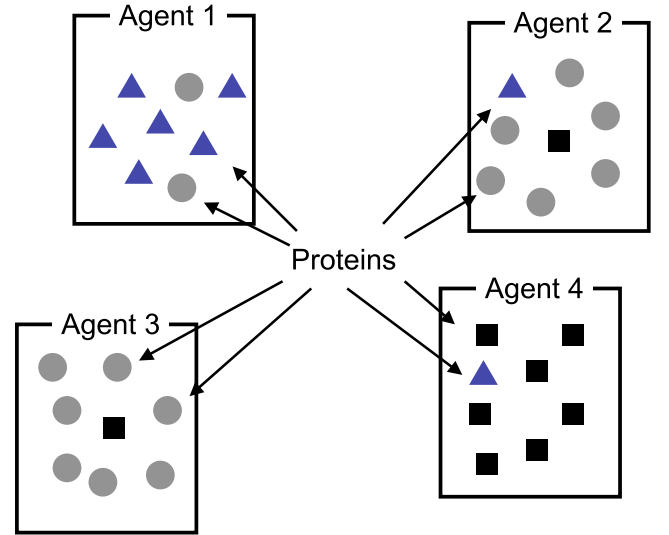

Figure 2: Type Agents. Each type agent determines the protein types in its cluster. Each agent makes decisions about the type of protein in its cluster independently, so proteins in the same cluster can have different types.

4. AGENT FRAMEWORK FOR PROTEIN MODEL

In this paper, agents play an important role in determining the specialization of the breaker proteins and determining the properties of the clusters. Specifically, each cluster will be assigned a "type" agent, whose actions determine the type of each protein in its cluster. Also each specialization

of a breaker protein will be determined by the actions of a "breaker-specialization" agents. This section describes the details of the learning process and the actions taken by the agents.

4.1 Type Agents

The types of the proteins in a cluster are determined by an agent assigned to that cluster. In our model an agent takes a discrete action ranging from 1 to k, where k is the number of possible types. Every time an agent takes an action, its action determines the type of a single protein in its cluster chosen at random. Over time, an agent takes many actions, defining the types of more and more proteins within the cluster. The goal of the agent is to choose types that ultimately lead to a protocell of high global utility. To accomplish this task, our agents learn through a simple reinforcement learning.

The system is modeled using a series of discrete time steps, where each agent takes an action after each time step. At the beginning of a time step each type agent takes an action determining the type of single protein in its cluster. This action is determined by an ϵ-greedy reinforcement learner with a learning table of size k, the number of protein types. Each entry in the table represents the agents expected utility when choosing the action associated with that entry. With probability $1 - \epsilon$ an agent takes an action associated with the highest table entry. With probability ϵ it takes a random action. Note that with the ϵ-greedy learner, we would expect a cluster to have proteins of several different types. Most of the time the agent would set proteins to the type associated with highest expected utility, though sometimes it will choose random types with frequency depending on ϵ.

If an agent sets the protein type to something other than a combiner protein or a breaker protein, then nothing happens in the system until the next time step. However, if an agent chooses a protein to be a combiner or a breaker protein, the corresponding combining or breaking operations then take place. If the protein is a combiner, then two other proteins are chosen at random and combined together to form a new protein. If the protein is a breaker protein, then another protein is chosen at random for possible breaking. Whether this protein is actually broken depends on the specialization of the breaker protein, discussed next.

4.2 Breaker-Specialization Agents

Each protocell has several breaker-specialization agents associated with the proteins. A breaker-specialization agent will typically be associated with several proteins, but a protein will only be associated with one breaker-specialization agent. The actions of its breaker-specialization agent determine the specialization of any breaker proteins that are associated with it. At each time step, the breaker-specialization agent takes one of k actions chosen with an ϵ-greedy reinforcement learner. This action then defines the specialization of a breaker protein chosen randomly. This specialization determines the type of protein the breaker protein cannot break: If the protein it operates on during its breaking operation is of the same type as its specialization, then the operation does nothing.

4.3 Cluster Assignment of New Proteins

After a combining or breaking operation, the cluster that the new protein belongs to is determined by the clusters of the original proteins involved in the operation. In our model we want the child protein(s) that result from these operations to end up in cluster(s) that are "similar" to the cluster(s) the parent protein(s) came from. While there are many ways to define the similarity between clusters, our model will use the learning tables for the agents assigned to each cluster to define similarity (Figure 3). For a breaking operation, a protein p in cluster c is broken into two new proteins. To find the cluster c_{new} for one of the new proteins p_{new}, we first compute a vector, v as follows:

$$v = Q_c + e \; , \qquad (2)$$

where Q_c is the learning table of the type agent associated with cluster c, and e is Gaussian noise. The cluster for the new protein, c_{new} is then chosen as the cluster associated with the agent that has the closest learning table to v, based on the Manhattan Norm (1-norm):

$$c_{new} = argmin_c \delta(Q_c, v) \; , \qquad (3)$$

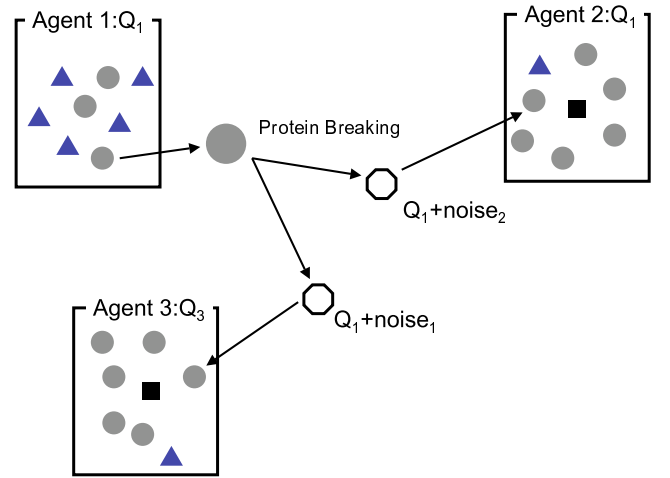where $\delta(\cdot)$ is the Manhattan Norm.

Figure 3: **Breaking Proteins Apart. When a protein is broken apart, two child proteins are created. The types of each of the child proteins depend on the learning table of the agent for the cluster the original protein came from. For each child protein a new table is made by adding noise to this learning table. Each child protein is assigned to the the cluster whose agent has the learning table is closest to the proteins table.**

For a combining operation a protein p_1 in cluster c_1 is combined with protein p_2 in cluster c_2 to form a single new protein, p_{new}. As with the breaking operation, to find the cluster c_{new} for the new protein we first compute a vector, v. However this time v is a function of two learning tables since p_{new} is created from two proteins. To handle this we combine the learning tables using the element-wise max operator $m(x, y)$, which assigns to each component of m_i the maximum of x_i or y_i: $m_i = \max(x_i, y_i)$. The vector v can then be computed as:

$$v = m(Q_{c_1}, Q_{c_2}) + e \; , \qquad (4)$$

where Q_{c_1} and Q_{c_2} are the learning tables of the type agents associated with clusters c_1 and c_2, and e is Gaussian noise.

As before the cluster for the new protein, c_{new} is then chosen as the cluster associated with the agent that has the closest learning table to v, based on the Manhattan Norm (1-norm):

$$c_{new} = argmin_c \delta(Q_c, v) , \qquad (5)$$

where $\delta(\cdot)$ is the Manhattan Norm. The process for determining the breaker-specialization agent associated with a new protein is parallel, with the learning tables of the breaker-specialization agents being used.

4.4 Agent Utilities

The goal of the agents is to maximize the global utility $G(x, y)$ as defined in Equation 1. However when there are many agents, it is difficult to maximize this utility. This difficulty arises from the fact that a single agent will only have a small impact on the value of G. If an agent takes an action and G increases, that agent does not know if the increases were due to its action or the actions of the other agents. When agents do not have significant influence over their utility as compared to the influence of all the other agents, it usually takes a very long time for them to learn to take actions that maximize it.

To alleviate this problem, this paper has each agent maximize a "difference utility" instead of the global utility. A difference utility is specific to an agents i, and can be computed as follows:

$$D_i(z) = G(z) - G(z_{-i}) , \qquad (6)$$

where z is a vector of all the actions taken by all the agents and z_{-i} is a vector of all the actions except for the action taken by agent i. The second term of the difference equation is a counterfactual version of the global utility that can be seen as the value of the system without agent i. Therefore the difference utility can be seen as the agent's net contribution to the system. By subtracting this counterfactual second term from first term, the difference equation removes much of the noise caused by the actions of the other agents. The difference utility has been shown to greatly speed up reinforcement learning in many domains, including internet traffic routing, multi-robotic control and satellite communication [9, 10, 1].

The specific form of the difference utility for our global utility, $G(x, y)$, defined in this paper is:

$$D_i(x, y) = G(x, y) - G(x_{-i}, y) , \qquad (7)$$

where x_{-i} represents the counts of the protein types for a protocell, had agent i had not taken any action. The exact form of x_{-i} depends on the type of action made, especially if the agent made a protein a combiner or a breaker. For all actions let j be the type of protein chosen by the agent. For combiner actions let a and b be the types of proteins that were combined and let c be the type of protein produced. For breaker actions let a be the type of protein broken and let b and c be the types of proteins produced. Then the value of x_{-i} is as follows:

1. $x - e_j$ for non-combiner or breaker proteins ,

2. $x - e_j + e_a + e_b - e_c$ for combiner proteins ,

3. $x - e_j + e_a - e_b - e_c$ for breaker proteins ,

where e_j is a vector where element j has a value of 1 and the other elements are 0. Essentially for combiner and breaker proteins, the proteins created are removed from the counts and the proteins consumed are returned.

5. RESULTS

To test the ability of our agent-based system to produce effective models, we ran a number of experiments where agents had to create models that led to protocells with high utility starting with different protein distributions. In this task as agents had to create a model with the proper distribution of combiner and breaker proteins, and had to ensure that the breaker proteins had the proper selectivity profile. In addition agents had to ensure that the functions of all the proteins were in the proper distribution.

In all the experiments there were thirty agents. In each experiment there were five different types of proteins that were desired in the final outcome besides combiner and breaker proteins (e.g., proteins for energy collection, energy transport, membrane building, and nucleus building). The experiment tested different starting conditions, different desirable combinations of these proteins by selecting an optimal number of each these five proteins, y_1, y_2, y_3, y_4, y_5, along with a corresponding optimal total $y = y_1 + y_2 + y_3 + y_4 + y_5$, and changing optimal conditions.

- The first experiment tested the ability of the agents to create a model that reached a good final equilibrium, starting from y proteins. The intent of this experiment was to determine whether the desired combination of proteins can be reached by rearranging the proteins present at the start.

- The second experiment tested this ability starting from 2 proteins. The intent of this experiment was to determine how few proteins are needed for the process to start, and whether a protocell can develop starting from just two proteins and generate the necessary proteins to perform all the required tasks.

- The third experiment tested this ability starting with $3y$ proteins. The intent of this experiment was to test whether the protocell can cut through clutter and eliminate unnecessary proteins.

- The fourth experiment tested the ability of the protocell to adapt to a sudden change in environment where the optimal number of each type of protein is changed halfway through the simulation. The intent of this experiment was to see the robustness of the protocell to sudden changes in its environment.

5.1 Protocell Evolution from y Proteins

In the first experiment the protocell started with 30 proteins. The optimal number of proteins was 31 ($1+2+4+8+16$), with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. Since the starting and optimal number of proteins was almost the same, this problem was relatively simple and the only task of the agents was to create a model that forms the proper distribution and maintains it. However, despite the simplicity of the task, Figure 4 shows that agents directly maximizing the global utility could not produce an adequate model. In fact the initial actions of the agents produced a model that was worse than the starting one, and after 5000 learning trials they still did not manage to reach their initial starting performance. This result is not surprising since many choice

of model parameters can be very destructive, such as creating too many breaker proteins. Since it was very difficult for the agents to learn from the global utility, they had difficulty avoiding these destructive choices. In contrast, agents using the difference utility were able to produce adequate models for this task. By changing the distribution they were able to create a model that achieved a utility of 3.7, up from the initial utility of 2.8. The agents using the difference utility were more effective than the agents using the global utility because the difference utility was much more learnable. The significance of this model to non-genomic evolution, however is minimal. It shows that the proper distribution can be maintained, intuitively meaning that there should be few combiner and breaker proteins.
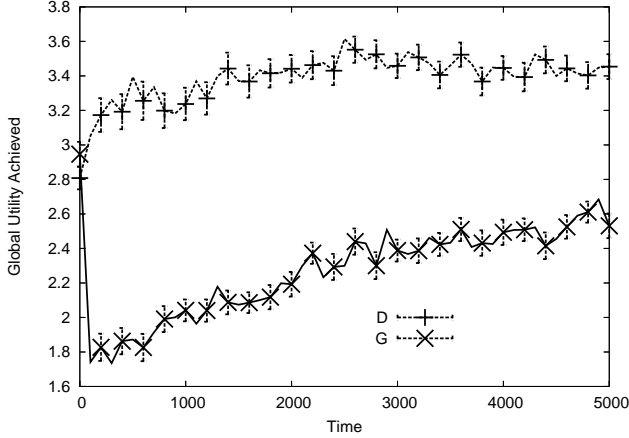

Figure 4: **Performance for protocell starting with near optimal number of proteins. Agents using difference utility form adequate model by changing protein distributions to fit targets. Global utility cannot reach original (random) performance.**

5.2 Protocell Evolution from 2 Proteins

In the second experiment, the protocell started with 2 proteins. The optimal number of proteins was 31, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. This presented a much more difficult setting than the previous one, since starting with 2 proteins, the agents had to create a model that increases the number of proteins by a factor of 15, while providing that the proteins are in the proper distribution. This requires an initial increase in the distribution of breaker proteins to create enough total proteins and a shift in the distribution, as well, to maintain the proper protein levels. Figure 5 shows that agents using the difference utility are able to create a model that leads to highly fit protocells. In this experiment, the performance of the protocell goes from 0.8 to 3.5, a significantly larger increase than in the previous experiment. This larger increase is expected since the initial protocell with two proteins is in a much worse state than the one starting with 30. This experiment shows that a model exists where a protocell with a small number of proteins can grow to one with a large number of proteins and form the proper protein distribution. This has significant insight into the required initial conditions for the formation of protocells. This result demonstrates that with the proper conditions a protein ecology can be formed starting from only 2 proteins.
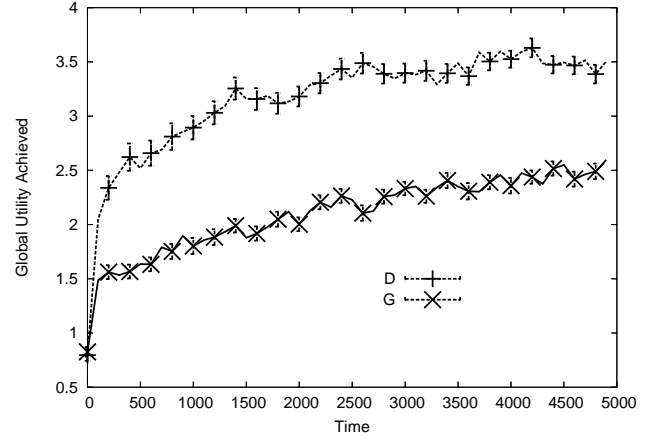

Figure 5: **Performance for protocell starting with 2 proteins. Agents using difference utility are able to form model where combiner and breaker proteins are formed, and lead protocell to an equilibrium where there are enough proteins, and the proteins are in the correct distribution.**

5.3 Protocell Evolution from 3y Proteins

In the third experiment the protocell started with 95 proteins. The optimal number of proteins was again 31, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8$, and $y_5 = 16$. This experiment requires an initial increase in the distribution of combiner proteins so that proteins in the initial high population are combined together to form a smaller population. As before results show that agents using the difference utility are able to create a model that leads to highly fit protocells. In this experiment the performance of the protocell quickly shoots from 2.2 to 3.6, but then goes down slightly. This slight reduction in utility is due to the model producing too many combiner proteins that causes the system to overshoot its protein reduction. Figure 6 shows that when there are too many proteins in a protocell, a mechanism can be created to reduce the number and create a proper protein distribution.

5.4 Protocell Evolution in Dynamic Environments

In the final experiment, the protocell started with 35 proteins with an optimal number of proteins of 31. The optimal number of proteins was thirty one, with $y_1 = 1, y_2 = 2, y_3 = 4, y_4 = 8, y_5 = 16$, but at 5000 episodes is changed to $y_1 = 16, y_2 = 8, y_3 = 4, y_4 = 2$, and $y_5 = 1$. This experiment requires the agents to adapt to new model requirements after they have already developed an existing model. The results shown in Figure 7 show that the agents are able to quickly adapt to this change in model requirements.

6. DISCUSSION AND CONCLUSION

In this paper we explore multi-agent models to simulate potential ways in which early non-genomic cells can evolve to perform the tasks required for survival. Through the use of learning agents, we create an abstract dynamical model
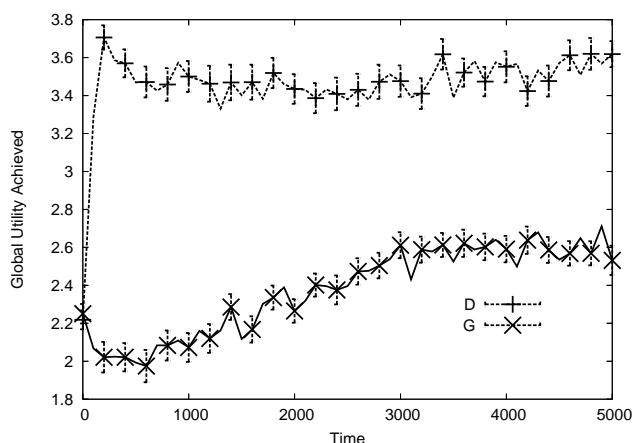
Figure 6: Performance for protocell starting with three times too many proteins. Agents using difference utility are able to form model where combiner and breaker proteins are formed that lead protocell to an equilibrium where there are enough proteins, and the proteins are in the correct distribution.
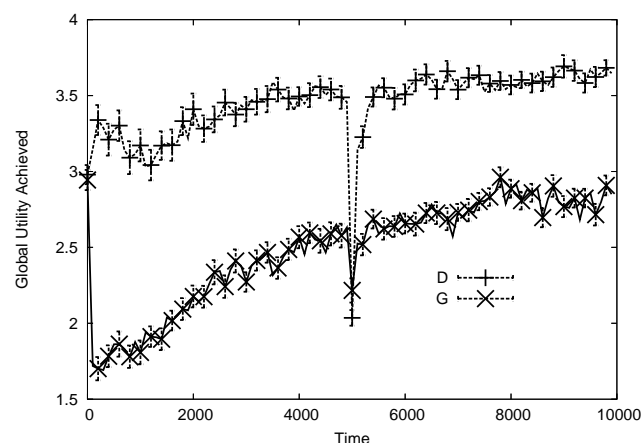

Figure 7: Model Formation for Changing Distribution. When requirements for the distribution of proteins changes at time step five thousand, agents are able to form a model that adjust to change.

for a primitive protocell of considerably greater complexity than previous models. In particular our models include proteins that had many different properties and had more complex dynamics for protein combining and breaking. In addition, while the fitness utility used in other models directly reflects the dynamics of the system, such as efficiency, our fitness utility is only indirectly influenced by the dynamics. Therefore our fitness utility is more realistic and provides many more degrees of freedom for the system to evolve. With our framework, we show that models exists that can produce highly fit protocells from a wide variety of starting conditions.

While our use of multi-agent systems allow us to create models of non-genomic evolution that are more complex than previous models, they are still highly abstract. One issue with the model is that only combiner and breaker proteins affect the dynamics of the system. We are currently working with astrobiologists to help add other dynamics into the framework. Another issue is that the optimal protein distributions used in our global utility of the protocell are somewhat arbitrary. However our framework allows other distributions to be plugged in, and we are also working with astrobiologists to come up with more realistic values.

Even though many details need to be added to our models to increase their biological plausibility, the multi-agent framework provides the power and flexibility to do so and naturally matches the distributed properties of protein-only ecologies. In general more complex evaluation of protocells can simply be implemented through the agent utilities. In addition more complexity can be introduced by adding different types of agents or changing the agents action space. Due to their flexibility we believe that multi-agent systems will become an important tool the study of astrobiology.

7. REFERENCES

[1] A. Agogino and K. Tumer. Efficient evaluation functions for multi-rover systems. In *The Genetic and Evolutionary Computation Conference*, pages 1–12, Seatle, WA, June 2004.

[2] A.J. Hager, J.D. Pollard, and J.W. Szostak. Ribozymes: Aiming at RNA replication and peptide synthesis. *Chemistry and Biology*, 3:717–725, 1996.

[3] G.F. Joyce. Ribozymes - building the RNA world. *Current Biology*, 6:965–967, 1996.

[4] D.H. Lee, J.R. Granja, J.A. Martinez, K. Severin, and M.R. Ghadiri. A self-replicating peptide. *Nature*, 382:525–528, 1996.

[5] D.H. Lee, K. Severin Y. Yokobayashi, and M.R. Ghadiri. Emergence of symbiosis in peptide self-replication through a hypercyclic network. *Nature*, 390:591–594, 1997.

[6] M. H. New and Andrew Pohorille. An inherited efficiencies model of non-genomic evolution. *Simulation Theory and Practice*, 8:199–208, 2000.

[7] R.W. Roberts and J.W. Szostak. RNA-peptide fusions for the in vitro selection of peptides and proteins. In *Proceedings of the National Academy of Science USA*, pages 12297–12302, 1997.

[8] K. Severin, D.H. Lee, J.A. Martinez, M. Vieth, and M.R. Ghadiri. Dynamic error correction in autocatalytic peptide networks. *Angewandte Chemie International Edition*, 37:126–128, 1998.

[9] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[10] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[11] S. Yao, I. Ghosh, R. Zutshi, and J. Chmielewski. Selective amplification by auto and cross-catalysis in a replicating peptide system. *Naturre*, 396:447–450, 1998.